WHAT IS CLAIMED IS:

1. A method for determining instruction boundaries of at least one method body within a computer code loaded into a memory of a smart card comprising:

a) examining in a sequential manner each instruction of the at least one method body starting with a first instruction of the at least one method body for an instruction selected from a group consisting of a forward jump instruction and a valid ending instruction;

b) maintaining a FLR Pointer corresponding to the instruction of the at least one method body for which the farthest forward jump instruction or the farthest valid ending instruction is detected; and

c) terminating the examining for a forward jump or a valid ending instruction when the instruction under examination is beyond the instruction corresponding to the FLR Pointer.

2. The method of claim 1 further comprising:

d) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

3. The method of claim 2 wherein each of the steps a), b), c), and d) are performed on each of a successive method bodies of the at least one method body.

4. The method of claim 1 further comprising:

d) resolving each unresolved reference in each instruction of the at least one method body starting with the first instruction of the at least one method body and ending with the instruction corresponding to the FLR Pointer.

5. The method of claim 4 further comprising:

e) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

6. The method of claim 5 wherein each of the steps a), b), c), d), and e) are performed on each of a successive method bodies of the at least one method body.

7. The method of claim 1 further comprising:

d) determining if an exception handler code corresponding to the at least one method body is present; and

e) responsive to determining that an exception handler code corresponding to the at least one method body is present beyond the instruction corresponding to the FLR Pointer:

(i) examining in a sequential manner each instruction of the exception handler code corresponding to the at least one method body starting with a first instruction of the exception handler code corresponding to the at least one method body for an instruction selected from a group consisting of a forward jump instruction and a valid ending instruction;

(ii) maintaining a FLR Pointer corresponding to the exception handler code corresponding to the at least one method body for which the farthest forward jump or the farthest valid ending instruction is detected; and

(iii) terminating the examining for a forward jump or a valid ending instruction when the instruction under examination is beyond the instruction corresponding to the FLR Pointer.

8. The method of claim 7 further comprising:

f) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

9. The method of claim 8 wherein each of the steps a), b), c), d), e), and f) are performed on each of a successive method bodies of the at least one method body.

10. The method of claim 7 further comprising:

f) resolving each unresolved reference in each instruction of the exception handler code corresponding to the at least one method body starting with the first instruction of the exception handler code

corresponding to the at least one method body and ending with the instruction corresponding to the FLR Pointer.

11. The method of claim 10 further comprising:

g) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

12. The method of claim 11 wherein each of the steps a), b), c), d), e), f), and g) are performed on each of a successive method bodies of the at least one method body.

13. The method of claim 7 wherein determining if an exception handler code corresponding to the at least one method body is present includes:

(i) determining if an exception entry is available in an exception handler array corresponding to the computer code;

(ii) responsive to determining that an exception entry is available, determining a catch range for the exception entry; and

(iii) determining if any of the instructions of the at least one method body starting with the first instruction of the at least one method body and ending with the instruction corresponding to the FLR Pointer are within the catch range.

14. The method of claim 1 wherein the computer code comprises a methods item of a method component of a converted applet file.

15. The method of claim 1 wherein the memory comprises non-volatile read/write memory.

16. A computer-readable medium tangibly having a program of machine-readable instructions for causing a processor to perform a method for determining instruction boundaries of at least one method body within a computer code loaded into a memory of a smart card, the method comprising:

a) examining in a sequential manner each instruction of the at least one method body starting with a first instruction of the at least one method body for an instruction selected from a group consisting of a forward jump instruction and a valid ending instruction;

b) maintaining a FLR Pointer corresponding to the instruction of the at least one method body for which the farthest forward jump instruction or the farthest valid ending instruction is detected; and

c) terminating the examining for a forward jump or a valid ending instruction when the instruction under examination is beyond the instruction corresponding to the FLR Pointer.

17. The computer-readable medium of claim 16 further having instructions for causing a processor to perform a method, the method comprising:

d) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

18. The computer-readable medium of claim 17 further having instructions for causing a processor to perform each of the steps a), b), c), and d) on each of a successive method bodies of the at least one method body.

19. The computer-readable medium of claim 16 further having instructions for causing a processor to perform a method, the method comprising:

d) resolving each unresolved reference in each instruction of the at least one method body starting with the first instruction of the at least one method body and ending with the instruction corresponding to the FLR Pointer.

20. The computer-readable medium of claim 19 further having instructions for causing a processor to perform a method, the method comprising:

e) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

21. The computer-readable medium of claim 20 further having instructions for causing a processor to perform each of the steps a), b), c), d), and e) on each of a successive method bodies of the at least one method body.

22. The computer-readable medium of claim 16 further having instructions for causing a processor to perform a method, the method comprising:

d) determining if an exception handler code corresponding to the at least one method body is present; and

e) responsive to determining that an exception handler code corresponding to the at least one method body is present beyond the instruction corresponding to the FLR Pointer:

(i) examining in a sequential manner each instruction of the exception handler code corresponding to the at least one method body starting with a first instruction of the exception handler code corresponding to the at least one method body for an instruction selected from a group consisting of a forward jump instruction and a valid ending instruction;

(ii) maintaining a FLR Pointer corresponding to the exception handler code corresponding to the at least one method body for which the farthest forward jump or the farthest valid ending instruction is detected; and

(iii) terminating the examining for a forward jump or a valid ending instruction when the instruction under examination is beyond the instruction corresponding to the FLR Pointer.

23. The computer-readable medium of claim 22 further having instructions for causing a processor to perform a method, the method comprising:

f) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

24. The computer-readable medium of claim 23 further having instructions for causing a processor to perform each of the steps a), b), c), d), e), and f) on each of a successive method bodies of the at least one method body.

25. The computer-readable medium of claim 22 further having instructions for causing a processor to perform a method, the method comprising:

f) resolving each unresolved reference in each instruction of the exception handler code corresponding to the at least one method body starting with the first instruction of the exception handler code corresponding to the at least one method body and ending with the instruction corresponding to the FLR Pointer.

26. The computer-readable medium of claim 25 further having instructions for causing a processor to perform a method, the method comprising:

g) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

27. The computer-readable medium of claim 26 further having instructions for causing a processor to perform each of the steps a), b), c), d), e), f), and g) on each of a successive method bodies of the at least one method body.

28. The computer-readable medium of claim 22 wherein the instructions for determining if an exception handler code corresponding to the at least one method body is present includes instructions for causing a processor to perform a method, the method comprising:

(i) determining if an exception entry is available in an exception handler array corresponding to the computer code;

(ii) responsive to determining that an exception entry is available, determining a catch range for the exception entry; and

(iii) determining if any of the instructions of the at least one method body starting with the first instruction of the at least one method body and ending with the instruction corresponding to the FLR Pointer are within the catch range.

Patent Application

29. A smart card configured to receive computer code having at least one method body within the computer code comprising:

a memory;

a processor connected to the memory; and

an installer module having logic operable to cause the processor to receive the computer code into the memory; and further having logic operable to cause the processor to determine instruction boundaries within the computer code by a) examining in a sequential manner each instruction of the at least one method body starting with a first instruction of the at least one method body for an instruction selected from a group consisting of a forward jump instruction and a valid ending instruction; b) maintaining a FLR Pointer corresponding to the instruction of the at least one method body for which the farthest forward jump instruction or the farthest valid ending instruction is detected; and c) terminating the examining for a forward jump or a valid ending instruction when the instruction under examination is beyond the instruction corresponding to the FLR Pointer.

30. The smart card of claim 29 further having logic operable to cause the processor to determine instruction boundaries within the computer code by d) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

31. The smart card of claim 30 wherein each of the steps a), b), c), and d) are performed on each of a successive method bodies of the at least one method body.

32. The smart card of claim 29 further having logic operable to cause the processor to resolve unresolved references in the at least one method body by d) resolving each unresolved reference in each instruction of the at least one method body starting with the first instruction of the at least one method body and ending with the instruction corresponding to the FLR Pointer.

33. The smart card of claim 32 further having logic operable to cause the processor to determine instruction boundaries within the computer code by e) setting a Start

Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

34. The smart card of claim 33 wherein each of the steps a), b), c), d), and e) are performed on each of a successive method bodies of the at least one method body.

35. The smart card of claim 29 further having logic operable to cause the processor to determine instruction boundaries within the computer code by d) determining if an exception handler code corresponding to the at least one method body is present; and e) responsive to determining that an exception handler code corresponding to the at least one method body is present beyond the instruction corresponding to the FLR Pointer: (i) examining in a sequential manner each instruction of the exception handler code corresponding to the at least one method body starting with a first instruction of the exception handler code corresponding to the at least one method body for an instruction selected from a group consisting of a forward jump instruction and a valid ending instruction; (ii) maintaining a FLR Pointer corresponding to the exception handler code corresponding to the at least one method body for which the farthest forward jump or the farthest valid ending instruction is detected; and (iii) terminating the examining for a forward jump or a valid ending instruction when the instruction under examination is beyond the instruction corresponding to the FLR Pointer.

36. The smart card of claim 35 further having logic operable to cause the processor to determine instruction boundaries within the computer code by f) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

37. The smart card of claim 36 wherein each of the steps a), b), c), d), e), and f) are performed on each of a successive method bodies of the at least one method body.

38. The smart card of claim 35 further having logic operable to resolve unresolved reference in the exception handler code corresponding to the at least one method body by f) resolving each unresolved reference in each instruction of the exception handler code corresponding to the at least one method body starting with the first instruction

of the exception handler code corresponding to the at least one method body and ending with the instruction corresponding to the FLR Pointer.

39. The smart card of claim 38 further having logic operable to cause the processor to determine instruction boundaries within the computer code by g) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

40. The smart card of claim 39 wherein each of the steps a), b), c), d), e), f), and g) are performed on each of a successive method bodies of the at least one method body.

41. The smart card of claim 35 further having logic operable to cause the processor to determine if an exception handler code corresponding to the at least one method body is present by (i) determining if an exception entry is available in an exception handler array corresponding to the computer code; (ii) responsive to determining that an exception entry is available, determining a catch range for the exception entry; and (iii) determining if any of the instructions of the at least one method body starting with the first instruction of the at least one method body and ending with the instruction corresponding to the FLR Pointer are within the catch range.

42. The smart card of claim 29 wherein the computer code comprises a methods items of a method component of a converted applet file.

43. The smart card of claim 29 wherein the memory comprises non-volatile read/write memory.

44. A method for determining instruction boundaries of at least one method body within a computer code loaded into a memory of a smart card comprising:

a) examining the instructions of the at least one method body to determine a farthest logical return within the at least one method body;

b) establishing the instruction boundary at the instruction located at the farthest logical return; and

c) terminating the examination of the instructions when the instruction under examination is beyond the farthest logical return.

45. The method of claim 44 wherein the examining step (a) includes examining for an instruction selected from a group consisting of a forward jump instruction and a valid ending instruction.

46. The method of claim 44 further comprising:

d) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the farthest logical return.

47. The method of claim 46 wherein each of the steps a), b), c), and d) are performed on each of a successive method bodies of the at least one method body.

48. The method of claim 44 further comprising:

d) resolving each unresolved reference in each instruction of the at least one method body starting with the first instruction of the at least one method body and ending with the instruction corresponding to the farthest logical return.

49. The method of claim 48 further comprising:

e) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the farthest logical return.

50. The method of claim 49 wherein each of the steps a), b), c), d), and e) are performed on each of a successive method bodies of the at least one method body.

51. The method of claim 44 further comprising:

d) determining if an exception handler code corresponding to the at least one method body is present; and

e) responsive to determining that an exception handler code corresponding to the at least one method body is present beyond the instruction corresponding to the farthest logical return:

(i) examining the instructions of the exception handler code corresponding to the at least one method body to determine a farthest logical return within the exception handler code corresponding to the at least one method body;

(ii) establishing the instruction boundary at the instruction located at the farthest logical return within the exception handler code corresponding to the at least one method body; and

(iii) terminating the examination of the instructions when the instruction under examination is beyond the farthest logical return within the exception handler code corresponding to the at least one method body.

52. The method of claim 51 wherein the examining step (e)(i) includes examining for an instruction selected from a group consisting of a forward jump instruction and a valid ending instruction.

53. The method of claim 51 further comprising:

f) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the farthest logical return within the exception handler code corresponding to the at least one method body.

54. The method of claim 53 wherein each of the steps a), b), c), d), e), and f) are performed on each of a successive method bodies of the at least one method body.

55. The method of claim 51 further comprising:

f) resolving each unresolved reference in each instruction of the exception handler code corresponding to the at least one method body starting with the first instruction of the exception handler code corresponding to the at least one method body and ending with the

instruction corresponding to the farthest logical return within the exception handler code corresponding to the at least one method body.

56. The method of claim 55 further comprising:

g) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the farthest logical return within the exception handler code corresponding to the at least one method body.

57. The method of claim 56 wherein each of the steps a), b), c), d), e), f), and g) are performed on each of a successive method bodies of the at least one method body.

58. The method of claim 51 wherein determining if an exception handler code corresponding to the at least one method body is present includes:

(i) determining if an exception entry is available in an exception handler array corresponding to the computer code;

(ii) responsive to determining that an exception entry is available, determining a catch range for the exception entry; and

(iii) determining if any of the instructions of the at least one method body starting with the first instruction of the at least one method body and ending with the instruction corresponding to the farthest logical return are within the catch range.

59. A smart card configured to receive computer code having at least one method body within the computer code comprising:

a memory;

a processor connected to the memory; and

an installer module having logic operable to cause the processor to receive the computer code into the memory; and further having logic operable to cause the processor to determine instruction boundaries within the computer code by a) examining the instructions of the at least one method body to determine a

farthest logical return within the at least one method body; b) establishing the instruction boundary at the instruction located at the farthest logical return; and c) terminating the examination of the instructions when the instruction under examination is beyond the farthest logical return.

60. The smart card of claim 59 wherein the computer code comprises a methods item of a method component of a converted applet file.

61. The method of claim 44 wherein the memory comprises non-volatile read/write memory.

62. A method for determining instruction boundaries of at least one method body within a computer code loaded into a memory of a smart card comprising:

(a) examining in a sequential manner each instruction of the at least one method body starting with a first instruction of the at least one method body for an instruction that establishes a logical return;

(b) maintaining a FLR Pointer to continuously store the farthest logical return found in the examining step (a); and

(c) terminating examination of the instructions when an instruction under examination is beyond the instruction corresponding to the FLR Pointer.

63. The method of claim 62 wherein the examining for an instruction that establishes a logical return of step (a) includes examining for an instruction selected from a group consisting of a forward jump instruction and a valid ending instruction.

64. The method of claim 63 further comprising:

d) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

65. The method of claim 64 wherein each of the steps a), b), c), and d) are performed on each of a successive method bodies of the at least one method body.

66. The method of claim 62 further comprising:

d) resolving each unresolved reference in each instruction of the at least one method body starting with the first instruction of the at least one method body and ending with the instruction corresponding to the FLR Pointer.

67. The method of claim 66 further comprising:

e) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

68. The method of claim 67 wherein each of the steps a), b), c), d), and e) are performed on each of a successive method bodies of the at least one method body.

69. The method of claim 62 further comprising:

d) determining if an exception handler code corresponding to the at least one method body is present; and

e) responsive to determining that an exception handler code corresponding to the at least one method body is present beyond the instruction corresponding to the FLR Pointer:

(i) examining in a sequential manner each instruction of the exception handler code corresponding to the at least one method body starting with a first instruction of the exception handler code corresponding to the at least one method body for an instruction that establishes a logical return;

(ii) maintaining a FLR Pointer to continuously store the farthest logical return found in the examining step e(i);

(iii) terminating examination of the instructions when an instruction under examination is beyond the instruction corresponding to the FLR Pointer.

70. The method of claim 69 wherein the examining step (e)(i) includes examining for an instruction selected from a group consisting of a forward jump instruction and a valid ending instruction.

71. The method of claim 69 further comprising:

f) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

72. The method of claim 71 wherein each of the steps a), b), c), d), e), and f) are performed on each of a successive method bodies of the at least one method body.

73. The method of claim 69 further comprising:

f) resolving each unresolved reference in each instruction of the exception handler code corresponding to the at least one method body starting with the first instruction of the exception handler code corresponding to the at least one method body and ending with the instruction corresponding to the FLR Pointer.

74. The method of claim 73 further comprising:

g) setting a Start Pointer to a next method body of the at least one method body following the instruction corresponding to the FLR Pointer.

75. The method of claim 74 wherein each of the steps a), b), c), d), e), f), and g) are performed on each of a successive method bodies of the at least one method body.

76. The method of claim 69 wherein determining if an exception handler code corresponding to the at least one method body is present includes:

(i) determining if an exception entry is available in an exception handler array corresponding to the computer code;

(ii) responsive to determining that an exception entry is available, determining a catch range for the exception entry; and

(iii) determining if any of the instructions of the at least one method body starting with the first instruction of the at least one

method body and ending with the instruction corresponding to the FLR Pointer are within the catch range.

77. A smart card configured to receive computer code having at least one method body within the computer code comprising:

a memory;

a processor connected to the memory; and

an installer module having logic operable to cause the processor to receive the computer code into the memory; and further having logic operable to cause the processor to determine instruction boundaries within the computer code by a) examining in a sequential manner each instruction of the at least one method body starting with a first instruction of the at least one method body for an instruction that establishes a logical return; b) maintaining a FLR Pointer to continuously store the farthest logical return found in the examining step (a); and c) terminating examination of the instructions when an instruction under examination is beyond the instruction corresponding to the FLR Pointer.

78. The smart card of claim 77 wherein the computer code comprises a methods item of a method component of a converted applet file.

79. The method of claim 77 wherein the memory comprises non-volatile read/write memory.

Oberthur - Exhibit 1002